Лекция 2:

# **Модель COM:**

\*\*COM\*\* — пример спецификации для разработки слабосвязанных компонентный приложений.

\*\*COM-программисты\*\* — разрабатывают компоненты

Задача майкрософт: они хотели разработать стандарт программного кода, который представлял собой компоненту. Пример: когда таблицу из ворд можно перенести в excel.

Браузер тоже набор компоненты, сделанных по спецификации COM.

В основе COM-спецификации лежит модель программного обеспечения.

COM(Common Object Model).

Каждый объект COM имеет свой ID типа GUID(128 бит, никогда не повторяется)

#pragma once

#include <objbase.h>

GUID CLSID\_CA = (0x3f5d…)

COM объекты размещенны в контейнерах(DLL, EXE-module)

Объект + контейнер = COM-server.

DLL — объект, который работает в процессе, работающем в процессе другого приложения

DLL имеет свойство — загрузка в единственном экземляре. Если пытаются подключить второй раз, произойдет проекцировании библиотеки в ОЗУ других процессов.

Образ — часть сердца

Однокомпонентный сервер: один тип объекта

Многокомпонентный сервер: хранит несколько типов объктов

Идентификатор соответствует типу COM-объекта

Вся поддержка COM-технологии осуществляется спец.библотекой olea32

Для работы с библиотекой используется #include <objbase.h>

COM-Сервер типы:

* CLSCTX\_SERVER(DLL внутрипроцеснный сервер)
* CLSCTX\_LOCAL\_SERVER(EXE-сервер за границами процесса, гл та том же компьютере);
* CLSCTX\_REMOTE\_SERVER (EXE-сервер на удаленном компьютере);

COM-клиент – программ которая обращается к COM серверу , вызывая функцию, которая обращается к КОМ серверу, обращаясь к библиотеки ole32 , которая создает COM-объект

Задаем параметры и получаем результат

COM клиент 🡪 вызов функции OLE32.dll 🡪 COM-сервер 🡪 возврат управления в OLE32 🡪 возврат управления в COM-клиент

Все функции OLE32.dll возвращают значение типа HRESULT.

HRESULT – 32 битное число, в котором биты имеют свою структуру

Первые два бита — ошибка,

Дальше — резерв

Дальше —

Для того, чтобы работать с экземплярами, нам нужны интерфейсы

Для работы с объектом, должны знать SID и наборы интерфейсов(для вызова функций этого объекта)

Один COM-объект модет иметь один или несколько интерфейсов

Каждый интерфейс характеруется своим ID(GUID)

IUnknown объявлен в <Unknwn.h>

Его ID известен, зашит в заголовок и известна структура этого интерфейса(наборы функций, которые мы можем вызвать)

В IKnown три метода:

- QueryInterface(REFIID riid, void \_RPC\_FAB \*\*ppyObject) Возвращает HRESULT (соглашение о свзяи)

- AddRef(void0 – исключения, не возращают HRESULt

- Release(void)

QueryInterface для получения указатель на другой интерфейс

AddRef специальный метод, который позволяет увеличить счетчик обращения к интерфейсу.

Дело в том, что работы с компонентой, ведется счестчик

Release – уменьшает счетчик на 1

Если состояние > 0, то интерфейс используется

IUnKnown замечателен тем, что его все знают, что любой другой интефрейс в COM наследует IUKnown.

Аргументы передаются через стек, очистку стека производят вызываемые программы

Для того, чтобы Com клиент смог работать с COM-сервером он должен знать **CLSSID , тип сервера, идентификаторы интерфейсов.**

createInstance создает COM-объект

Параметры:

1) GUID

2) NULL

3) Тип сервера

4) Какой я хочу интерфейс запросить

5) Указатель на запрашиваемый интерфейс

1) Local exe

2) Удаленный exe

3) In proc(dll)

Кеширование – из менее быстрой памяти в более быструю память